Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Janez Štok

Koncept informacijskega sistema rezervacij

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Borut Batagelj

Ljubljana, 2016

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V svojem delu razvite prototip informacijskega sistema za rezervacije. Sistem naj omogoča enotno platformo s pomočjo katere bodo lahko ponudniki gostinskih storitev ponudili svojo hrano in prostor. Uporabnikom naj bo omogočeno, da bodo izvedli rezervacijo tudi preko mobilne naprave. Pri načrtovanju sistema natančno opišite vse uporabljene tehnologije in postopek izdelave takšnega sistema rezervacij.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Janez Štok sem avtor diplomskega dela z naslovom:

Koncept informacijskega sistema rezervacij (angl. *Concept of restaurant reservations information system*)

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Boruta Batagelja,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 15. februarja 2016 Podpis avtorja:

Zahvaljujem se mentorju viš. pred. dr. Borutu Batagelju za pomoč in nasvete pri izdelavi diplomskega dela. Obenem gre zahvala tudi družini, ki mi je stala ob strani tekom študija in izdelavi diplomskega dela.

Kazalo

Slike

Seznam uporabljenih kratic

kratica	angleško	slovensko
AJAX	Asynchronous JavaScript and XML	Asinhroni Javascript in XML
API	Application Programming Interface	Vmesnik za programiranje aplikacij
CSS	Cascading Style Sheets	Kaskadne stilske podloge
HTTP	Hypertext Transfer Protocol	Protokol za prenos nadbesedil
HTML	HyperText Markup Language	Jezik za označevanje nadbesedila
IDE	Intergrated Development Enviroment	Integrirano razvojno okolje
JSON	JavaScript Object Notation	Notacija objektov Javascript
JVM	Java Virtual Machine	Javanski navidezni stroj
GUI	Graphical user interface	Uporabniški grafični vmesnik
SDK	Software Development Kit	Programski razvojni paket
SQL	Script Query Language	Strukturirani povpraševalni jezik
XML	Extensible Markup Language	Razširljiv označevalni jezik
URL	Uniform resource locator	Enolični krajevnik vira

Povzetek

Naslov: Koncept informacijskega sistema rezervacij

V domačem kraju imamo le eno gostilno, ki jo obiskujejo večinoma italijanski gosti. Časi so se spremenili in promet italijanskih gostov je upadel. Nekateri krajani so se odločili, da bi sobe oddajali prehodnim gostom. Gostje se velikokrat odločijo prav za to gostilno, saj se lahko do gostilne sprehodimo. Ker so gostje večinoma tujci, je uporaba tujega jezika skoraj obvezna. Lastnik gostilne razloži težavo. Potreboval bi mobilno aplikacijo, ki bi predstavila jedi s sliko, urejanje pa bi potekalo preko spletne strani. Ideja je zanimiva, a aplikacija je omejena samo na eno restavracijo oziroma gostilno. Porodi se ideja o informacijskem sistemu, ki bi omogočal registracijo več restavracij. Te bi predstavljale svoje jedi in imele možnost rezervacije na določeno jed ali mizo. V prvem delu diplomske naloge so opisane uporabljene tehnologije, med njimi Java, Javascript, PHP, HTML, CSS, MySQL, SQLite in JSON, sledijo pa razvojna orodja, med katerimi najdemo Android Studio in DBDesigner. V glavnem delu bo opisan potek izdelave aplikacije, uporaba tehnologij in uporabniških vmesnikov, kot arhitekture sistema. V zadnjem delu so podana sklepna opažanja in možnosti nadgradnje prototipa.

Ključne besede: restavracija, rezervacije, mobilna aplikacija, spletna aplikacija, Android.

Abstract

Title: Concept of restaurant reservations information system

In our home place we have only one inn. It's occupied mostly from Italian guests. Times are changing and buying power of Italians has declining since world financial collapse. Some fellow villagers has got an idea to rent some rooms for tourists. Inn is popular choice among guests, because is in walking distance of rooms. There is a problem with knowledge of languages, with that in mind owner of inn decided to contact me and explain the problem. There is need to create some sort of Android application witch would showcase dishes with some pictures. Customization will be made from website. Idea in general is interesting but is limited to just one restaurant or inn. So I came up with an idea of information system that is not limited to one restaurant, and I added book a table or book a dish. In the first chapter of paper I will explain used technologies, some of them are Java, Javascript, PHP, HTML, CSS, MySQL, SQLite and JSON, following development tools Android Studio and DBDesigner. In he main chapter I will explain the flow development of app, use of technologies and GUI. In the last chapter I will say something about final thoughts and future possibility of upgrade.

Keywords: restauration, reservations, mobile application, web application, Android.

Poglavje 1

Uvod

Številne restavracije imajo informacijske sisteme, ki jim vodijo zalogo, račune in naročila, vendar so ti sistemi omejeni in prilagojeni točno določeni restavraciji. OpenTable [1] je sistem, ki omogoča rezervacijo s predstavitvene strani, vendar ne omogoča rezervacije jedi. Le-ta deluje tako, da se z iskanjem mize izdela seznam restavracij, ki so tisti termin proste in so v sistemu. Ostane še odločitev, kje bomo rezervirali termin. Sistem ima možnost pregleda tudi drugih terminov, ki so na voljo v naslednjih urah. Podjetje OpenTable je bilo ustanovljeno leta 1998. Omejeno je na Ameriko, Nemčijo in Veliko Britanijo. Podobni sistemi so: dimmi[2], MyTable[3], ZAGAT[4] in številni drugi, vsem pa je skupno da Slovenije ne pokrivajo. Dimmi je podružnica TripAdvisorja, stran ki je specializirana za rezervacije v Avstraliji. Deluje po enakem principu kot OpenTable. MyTable je zanimiv projekt, ki omogoča rezervacije obrokov, ki so servirani doma. Stran je omejena le na Los Angeles. Je edina, pri kateri smo zasledili da ima možnost mobilne aplikacije. Ta projekt, bi lahko uporabili kot osnovo za dodatno ponudbo v prihodnosti. Zagat sodeluje z OpenTable za rezervacijo jedi, ima pa zanimiv sistem iskanja, kjer izbiramo lokacijo in nam ponudi vse restavracije z ocenami hrane, cenovnega razreda, servisa in podobe restavracije.

V Sloveniji so restavracije po večini razdeljene v tri kategorije rezervacij. Delijo se na tiste, ki nimajo spletne strani in je mogoče rezervirati samo

preko telefonske številke, ki jo poiščemo v imeniku ali je napisana na letaku. Naslednja kategorija so restavracije, ki imajo svoje spletišče, na katerem pa nimajo implementiranega nikakršnega sistema za rezervacijo. Te imajo pogosto prikazano številko ali elektronski naslov za povpraševanje. Nekaj pa jih le ima svoj sistem, ki je v večini primerov v obliki kontaktnega obrazca oziroma povpraševanja. Tukaj podamo svoje podatke uro in datum rezervacije, število oseb ter en kontaktni podatek. To je v večini primerov elektronska pošta ali telefonska številka. Ko oddamo povpraševanje ta rezervacija ni potrjena, dokler nas predstavnik restavracije ne obvesti nazaj. Svetla izjema je na primer restavracija Manna [5] iz Ljubljane, ki ima spletno rezervacijo prek ponudnika "youcanbook.me" [6], a nimamo možnosti pregleda več restavracij na enkrat. Prej omenjeni sistem deluje, tako da v sklopu strani ponudimo povezavo do sistema za restavracijo, kjer so spisani vsi prosti in zasedeni termini rezervacije. Na podlagi tega se odločimo, kateri prost termin bomo izbrali. "youcanbook.me" ponuja univerzalno rešitev vseh vrst rezervacije. Sistem je univerzalen, zato ima svoje pomanjkljivosti, na primer da, ga ne moremo integrirati v svojo stran. Najpomembnejša pomanjkljivost je ta, da smo omenjeni s prikazom podatkov o restavraciji. V današnji dobi ne moremo mimo družabnih omrežij. Veliko slovenskih restavracij ima namreč na družabnih omrežjih svojo stran, ki je pogojno primerna za rezervacijo, saj dobimo z veliko frekvenco ljudi svojo bazo potencialnih strank, a tu spet nimamo sistema rezervacij. Dodatna vrednost je povratna informacija strank. Povpraševanje je še vedno preko sporočila ali telefonske številke.

Zaradi hitrega sodobnega življenjskega sloga ne moremo sedeti in čakati na izdajo jedi eno uro ali več. Danes mora biti hrana sveža, saj bi se v nasprotnem primeru restavracija hitro znašla na slabem glasu. To pa pomeni velik padec števila strank, na daljši rok pa tudi zaprtje restavracije. Vse pogosteje se tudi srečujemo s težavo prevelike količine odpadkov s strani prehrambne industrije. United Nations Environment Programme je objavil, da v razvitejših predelih sveta vsako leto odvržemo dvesto dvaindvajset milijonov ton hrane [7]. Prebivalstvo, ki je ekološko usmerjeno se lahko nad

tem podatkom samo zgrozi. Vse restavracije zagotovo ne delujejo na način, da si ustvarijo preveč zaloge, a to dela večina. Prehrambna industrija, bi zato potrebovala korenito spremembo. Ne nazadnje je to tudi v interesu restavracije in njenega osebja. Ker bi bila hrana optimizirana, bi to lahko opazili tudi na finančnem področju, kar bi pomenilo večji zaslužek za restavracijo ali nižje cene za stranke. Odločitev za temo iz tega področja je bila tako lažja. Z rezervacijo jedi se nam odpirajo možnosti optimizacije priprave jedi v restavracijah, kjer bi lahko bilo zanemarljivo malo odpadnih surovin. Načrtovanje bi se tako poenostavilo, da bi lahko hrano dobili že v nekaj minutah po vstopu v restavracijo. Pripravljena hrana, bi bila tako bolj zdrava in sveža ob tem pa okusnejša, poleg tega strankam ne bi bilo treba čakati na izdajo hrane. To pomeni več zadovoljnih strank in posledično večji zaslužek.

Verjetno se sprašujete, kako bi to lahko naredili? Odgovor, ki se sam ponuja, je v načrtovanju ustreznega podpornega informacijskega sistema. V restavraciji se usedemo za mizo in čakamo na natakarja, da nam prinese meni. Natakar nato vzame naročila za pijačo. Po približno petih minutah se vrne s pijačo in vzame naročila za naše jedi. Potem moramo čakati na izdajo, če je jed že vnaprej pripravljena, je velikokrat postana in pustega okusa, a jo dobimo prej. Vse jedi po naročilu je treba čakati tudi eno uro, hrana je bolj sveža in velikokrat okusnejša od vnaprej pripravljene. Življenjski slog nam narekuje, da želimo dobiti hrano, kar se da hitro. Velika večina se zato odloča za hitro prehrano, ki je slabša za zdravje.

Kako bi to spremenili? Potrebujemo hrano, ki je sveža in zdrava. Želimo si spremeniti, da nam je postrežena najhitreje. Rešitev je v sistemu, ki bi prevzel naročilo za jedi dan ali dva preden bi prišli v restavracijo. Zmanjšalo se bi čakanje na naročeno hrano, lahko tudi iz ene ure na vsega 15 minut. Kar je še pomembneje, vsa hrana je pripravljena za tisto trenutek in ne stoji v kuhinji nekaj ur ali več. Podoben sistem velja v nekaterih hotelih, ko ob večerji dobimo obrazec (natisnjen list papirja) na katerem obkrožimo, kaj bomo jedli jutri, seveda če smo nastanjeni več dni. Posledično je kakovost hrane na neprimerno višji ravni. Verjamem, da bi si tudi tisti, ki vsak dan

posegajo po hitri prehrani, iz tega razloga premislili, saj bi tako lahko dobili kakovostnejšo prehrano v enakem času, kot ga potrebujemo za pripravo hamburgerja ali podobne hitre prehrane. S tem skrajšamo čas izdaje kot tudi pripravo in načrtovanje zaloge.

Poglavje 2

Razvojna orodja in tehnologije

Razvoj mobilne in spletne aplikacije zahteva, da povezujemo več različnih tehnologij in orodij, ki delujejo v simbiozi kot celota. V naslednjem podpoglavju bomo predstavili tehnologije, to so JSON, PHP, JAVA ... Orodja, ki jih bomo spoznali pa so Android Studio, DBDesigner in Netbeans. Android studio, ki je nadomestil Eclipse, skrbi za razvoj mobilne aplikacije in nam omogoča, da aplikacijo tudi testiramo v posnemovalniku (angl. *emulator*). Za del, ki skrbi za podatkovno bazo smo uporabili DBDesigner, kjer smo ustvarili konceptualni model, nato smo s pomočjo programske funkcije izvozili skripto SQL. To smo kasneje naložili v strežniški sistem, ki ima naložen MySQL. Orodje Netbeans skrbi za urejanje spletnih dokumentov. Večina tehnologij je bila uspešno uporabljenih za izdelavo sistema za evidentiranje živine [8].

2.1 Tehnologije

2.1.1 AngularJS

AngularJS je odprtokodno ogrodje za spletne aplikacije, razvit s strani Miška Heverya in podjetja Google v letu 2009 [19]. Deluje po principu branja strani HTML, ki ima dodatne oznake za atribute. AngularJS interpretira oznake kot ukaze za metode, ki so napisane v AngularJS. To logiko bomo uporabili

za bolj čist "ajax" pristop pri obdelovanju form za spletni del aplikacije z uporabo JSON-a. Spet imamo lep primer, kjer se ne moremo izogniti uporabi le-tega.

2.1.2 Bootstrap

Bootstrap je brezplačna in odprtokodna zbirka orodij za ustvarjanje spletnih aplikacij in strani. Vsebuje HTML in CSS datoteke, ki so predloga dinamični spletni strani. Izvaja se na odjemalčevi strani (angl. *frontend*). Razvit je bil v organizaciji Twitter, kot ogrodje za zagotavljanje podpore več različnim dimenzijam prikazovalnika oziroma brskalnika. Ustanovitelja sta Mark Otto in Jacob Thornton z majhno ekipo. Podpirajo ga vsi vodilni brskalniki, tako mobilni kot namizni.

Od različice 2.0 je tudi prilagodljiv, kar pomeni, da se prilagaja več različnim dimenzijam prikazovalnika (Slika 2.1) [15]. Za izbiro Bootstrapa smo se odločili, da bi bila aplikacija optimizirana tudi za uporabo na mobilnih napravah. Z 21. 4. 2015 je iskalnik Google vpeljal, da bo v brskalniku na boljše mesto prišla stran, ki je narejena tudi za mobilne uporabnike, zato je razvoj s tem ogrodjem neizogiben [16].

Slika 2.1: Prikaz izgleda strani na več napravah.

2.1.3 CSS

CSS so podloge, ki v obliki preprostega slogovnega jezika skrbijo za videz spletnih strani. Z njimi definiramo slog dokumentov HTML elementov in

njihovo postavitev na strani. S tem dosežemo ločitev strukture strani z vsebino in njeno predstavitvijo. Dobimo večjo preglednost dokumentov HTML. Z značkami lahko uporabimo isto predlogo in s tem tudi omejimo ponavljanje kode. Predloge so se prvič pojavile leta 1994, ko so bile predstavljene na konferenci v Chicagu [14].

2.1.4 Java

Java je vse-opravilni objektno usmerjen programski jezik, ki je na trg prišel leta 1995, razvil pa ga je James Gosling v podjetju Sun Microsystem. Sintaksa programskega jezika izhaja iz starejših jezikov C in C++. Priljubljen je predvsem zaradi svoje objektne usmerjenosti. Pomembno je tudi omeniti, da je ta jezik brezplačen in odprtokoden, zato lahko vsakdo razvija v njem [13]. Možno ga je uporabljati v vseh napravah, kjer teče JVM, ker je JVM napisan za več operacijskih sistemov, je na voljo za veliko število različnih naprav.

2.1.5 JavaScript

Javascript je skriptni objektni programski jezik, ki ga je razvil Netscape za pomoč pri ustvarjanju interaktivnih spletnih strani. Jezik je zelo podoben programskem jeziku Java. Programi so navadno vključeni v dokument HTML. Izvaja se na odjemalčevi strani (front end). Spletnim stranem vdahne novo živahnost in uporabnost (spreminjanje slogov, barv, odzivi spletnih mest). Iz le tega so nastale številne knjižnice, kot je jQuery [12].

2.1.6 JSON

JSON je bil ustvarjen za povezavo med strežnikom in brskalnikom, brez uporabe Flasha ali Java appletov. Oseba zaslužna za današnjo specifikacijo, je Douglas Crockford. Opisan je tudi s strani dveh standardov, RFC 7159 in ECMA-404. ECMA standard opisuje samo določeno sintakso, medtem ko RFC ponuja tudi semantiko in varnostne napotke [9]. JSON se uporablja predvsem za povezavo med spletnim servisom in kakršno koli aplikacijo. V

našem primeru je uporabljena za povezavo med sistemom Android in bazo. JSON je predvsem bolj strjen kot XML, kar pomeni manj strukturiranih podatkov in posledično hitrejšo obdelavo za isto količino podatkov, zato je uporaba prvega hitrejša.

```
// Primer notacije
"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter","lastName":"Jones"}
]
```

Primer kode v formatu JSON [10]. Objekt "employees" ima tabelo s tremi vrsticami, ki vsebujejo objekte "firstName" in "lastName" z njihovimi vrednostmi.

2.1.7 MySQL

MySQL je odprtokoden sistem za upravljanje s podatkovnimi bazami relacijske baze. Za delo se uporablja prilagojeni jezik SQL. Deluje po principu odjemalec - strežnik. Del, ki skrbi za strežnik lahko namestimo na več neodvisnih strežnikov. Je tudi najbolj razširjen in standardizirani povpraševalni jezik. Trenutno je v lasti podjetja Oracle. Začetki segajo v leto 1995 [17]. Uporaba v naši aplikaciji je jasna. S pomočjo SQL CREATE stavkov smo bazo ustvarili z vsemi potrebnimi tabelami. Podatke smo pridobili ali spremenili z ukazi SELECT, UPDATE, DELETE. Strežnik php smo uporabili za odgovore v formatu JSON, ki jih je mobilna aplikacija zahtevala.

2.1.8 PHP

PHP (izvirno Personal Home Page Tools) je odprtokodni programski jezik, ki je prisoten pri razvoju dinamičnih spletnih strani [11]. Napisan je bil v programskem jeziku C. Oče sistema, je Rasmus Lerdorf. PHP teče na

spletnem strežniku. Izvorna koda je poslana na vhod, strežnik pa izpiše spletno stran kot izhod [11].

Slika 2.2: Primer spletnega servisa (tok podatkov)

Slika 2.2 prikazuje tok podatkov med strežnikom php, bazo in sistemom Android. Kot vmesni člen med sistemom Android in strežnikom pa uporabljamo JSON.

2.1.9 SQLite

Relacijska podatkovna baza, ki domuje na vgrajeni napravi in je zato najbolj primerna za shranjevanje lokalnih podatkov mobilne aplikacije. Ustvarjena je bila leta 2000 v pogodbi za ameriške marince (angl. *United States Navy*). Omogočala je zagon programa brez uporabe naprednih programov za upravljanje z bazo in brez administratorske prijave. To je pomembno predvsem zato, da se nam ni treba vedno znova prijavljati. Njena implementacija je zelo razvejana, podpirajo jo razni programi (Skype, brskalniki, CMS-i, itn. . .) in vsi vodilni mobilni operacijski sistemi [18]:

- Blackberry OS,
- Symbian ,
- Android,

- iOS,
- Windows 10,
- Tizen.

2.2 Razvojna orodja

2.2.1 Android Studio

Android Studio je Googlovo razvojno orodje za Android mobilne aplikacije. Od izdaje prve različice je bil mišljen kot nadomestek prejšnjega prilagojenega orodja Eclipse. Potreboval je vtičnike in knjižnice za delo z Androidom. Na primer, za delo z različnimi Googlovimi APIji smo včasih potrebovali različne knjižnice, ki smo jih uvozili v projekt, zdaj pa to ni potrebno. Izbira za naš projekt je bila samoumevna. Temelji na programskem jeziku Java in je izdan pod licenco Apache 2.0. Licenca Apache omogoča uporabo v nekomercialne in komercialne namene, zato omogoča, da razvijamo vse od zastonjskih do plačljivih ali naročniških aplikacij [20].

2.2.2 DBDesigner

Orodje omogoča načrtovanje podatkovnega modela, ki skrbi za shrambo in obdelavo podatkov. Najprej ustvarimo konceptualni model podatkovne baze, z ustreznimi entitetami in razmerji. Ne pozabimo na indekse in identifikacijske ključe. Nato lahko z orodjem izvozimo ukaze v poizvedovalnem jeziku SQL, tako da dobimo fizični model. Takega lahko uvozimo v našo podatkovno bazo.

2.2.3 Netbeans

Orodje Netbeans je še zadnje, ki ga bomo predstavili. Uporabljali ga bomo za urejanje PHP, HTML in CSS. IDE nam omogoča razvijanje za več programskih jezikov med drugimi C/C++, Java in prej omenjene. Temelji na

Javi in je zato primeren za večino popularnih platform, za katere je razvit JVM. Razvit je bil leta 1996, kot študentski projekt Xelfi Java IDE, Univerze v Pragi. Do leta 1999 je bil IDE plačljiv, po odkupu podjetja Sun Microsystems so ga izdali pod odprtokodno licenco in je brezplačen. Danes je to eno najboljših razvijalnih okolij [27].

2.2.4 Operacijski sistem Android

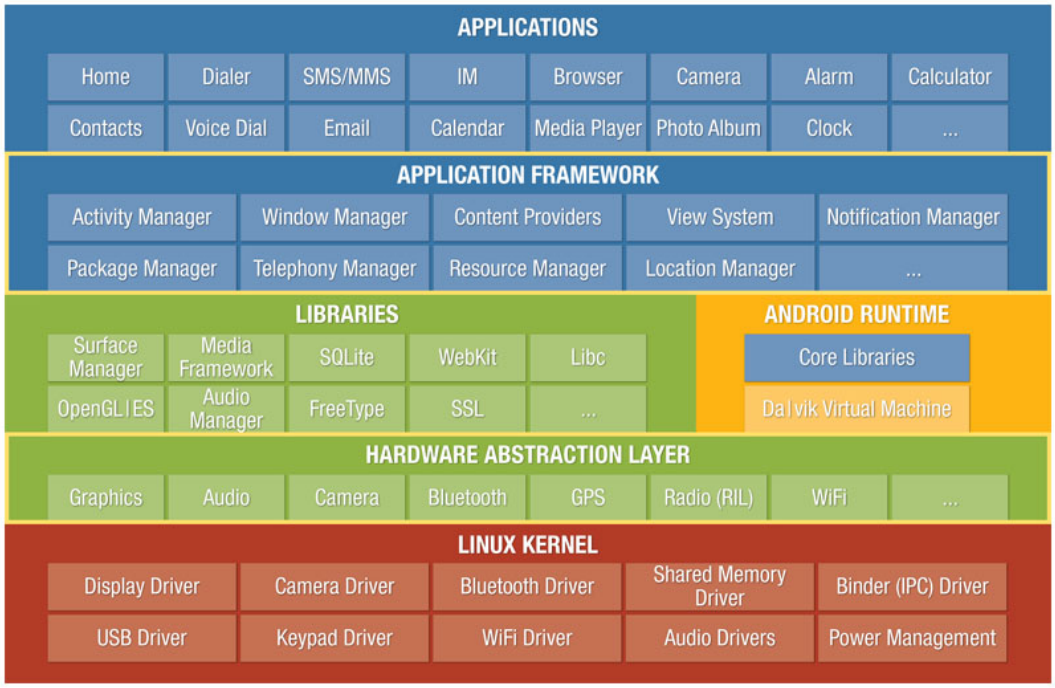
Ker gre razvoj spletnih sistemov v mobilno integracijo in optimizacijo, ne moremo mimo sistema Android. Mobilni operacijski sistem je namenjen tabličnim računalnikom in pametnim mobilnim telefonom. Po zadnjih podatkih ga uporablja 82,8 odstotkov pametnih telefonov, drugi z vsega 13,9 odstotki je Applov iOS. Če razvijemo aplikacijo za omenjena operacijska sistema, smo pokrili šestindevetdeset odstotkov uporabnikov. Omembe vredna sta še Windows Phone (Windows 10) in Blackberry OS [21].

Zasnovan je na jedru operacijskega sistema Linux. Google je leta 2005 prevzel podjetje Android Inc., nato se je začel skokovit razvoj operacijskega sistema, ki več kot uspešno traja do današnjih dni. Razvoj aplikacij poteka preko javanskega programskega jezika. [22]. Deluje preko optimiziranega aplikacijskega pogonskega sistema (angl. *Android runtime*), tako da aplikacijo pregleda in obdela njene podatke ob namestitvi, medtem ko starejša različica DALVIK pregleda le-to ob zagonu aplikacije. To se pozna predvsem ob večjih aplikacijah pri samem zagonu [23].

Ob končanem razvoju moramo ponuditi razvito aplikacijo svetu. Google in Apple imata posebne aplikacijske trgovine prav za to, da lahko razvijalci objavijo svoje izdelke. V trgovini najdemo vse vrste aplikacij (tako Apple kot Google imata na razpolago 1,5 milijona aplikacij). Za distribucijo aplikacije se moramo registrirati kot razvijalec. Potrebujemo uporabniški račun za razvijalce. Google račun za *Play store* stane v enkratnem plačilu 25 dolarjev, medtem ko Applov račun na leto stane 99 dolarjev.

2.2.4.1 Arhitektura sistema

Arhitektura Androida vsebuje šest komponent, ki skrbijo za nemoteno delovanje sistema. To so aplikacije, ogrodje aplikacij, zagonsko okolje, knjižnice in Linux jedro (Slika 2.3) [24].



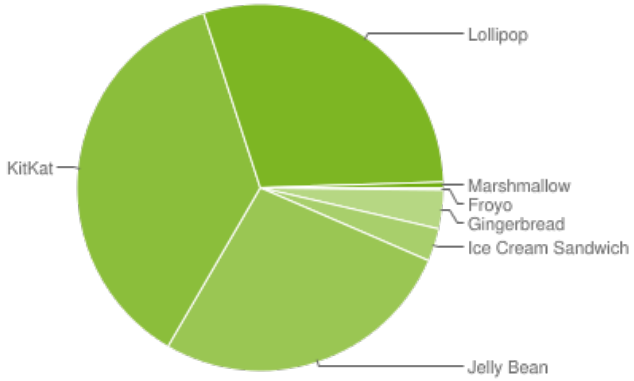
Slika 2.3: Diagram arhitekture sistema

Najpomembnejši del sistema je nedvomno Linuxovo jedro, ki zagotavlja varnost, upravljanje s pomnilnikom in procesi. Jedro skrbi za interpretacijo ukazov med strojno in programsko opremo. Dobre lastnosti so, da je hiter, zanesljiv in robusten.

2.2.4.2 Različice sistema

Ena največjih pomanjkljivosti oziroma za nekatere prednost sistema Android je razdrobljenost in več različnih "romov", ki se zelo razlikujejo od izvirne verzije. To so namreč prilagojene različice sistema Android, ki so jih proi-

zvajalci naprav prilagodili po svojih potrebah. Posledično se rok za izdajo nove verzije podaljša, lahko pa sploh ne podpirajo posodobitev, predvsem pri napravah nižjega in srednjega cenovnega razreda. Posledica tega je zelo velika razdrobljenost različic sistema, ki so aktivne. Prva različica sistema, ki jo je Google je leta 2009 razkril javnosti, se je imenovala Cupcake z oznako 1.5, pred dvema mesecema so izdali svežo različico Marsmallow z oznako 6.0. Razlike med najstarejšo in najnovejšo različico so prevelike, da bi razvijali in nudili podporo vsem sistemom. Moramo se odločiti, od katere različice naprej je razvoj še sprejemljiv. To se odločimo tako, da pregledamo, koliko različic je v uporabi in kakšne funkcije sistema bomo uporabljali. Naša aplikacija je združljiva z API različice 16, to pomeni, da je združljiva z Android Jelly Bean 4.1.x in naprej. Podpora zavzema slabih petindevetdeset odstotkov naprav. Spodnja slika 2.4 prikazuje aktivne različice sistema.



Slika 2.4: Različice sistema

Delež naprav od Froyo naprej [25]:

- Froyo 0,3%,
- Gingerbread 3,4%,
- Ice Cream Sandwich 2,9%,
- Jelly Bean 16,9%,

- KitKat 36,6%,
- Lollipop 29,5%,
- Marshmallow 0,5%.

Opazimo, da različice potrebujejo veliko časa, da si pridobijo delež med aktivnimi napravami. Vodilni KitKat je star dve leti, takoj za njim Lollipop, ki je bil izdan pred letom dni. Nov Marshmallow s komaj vidnim deležem je star dober mesec. Vsaka verzija tako potrebuje skoraj leto dni, da poseduje vidni delež na trgu. To je posledica razdrobljenosti sistema, ki smo ga omenili na začetku. Življenjska doba po priljubljenosti sistema je 2,5 - 3 leta, kar je pričakovana življenjska doba naprave.

2.2.5 PhpMyAdmin

PhpMyAdmin je brezplačno odprtokodno orodje za upravljanje MySQL podatkovne baze preko spletnega brskalnika. To je pomembno, saj lahko na preprost in pregleden način hitro urejamo na katerikoli napravi, ki podpira spletne strani pod pogojem, da je strežnik javno objavljen. Deluje od leta 1998 in ima do danes vodilno mesto po uporabi med ponudniki spletnih gostovanj [26].

Ključne funkcije orodja:

- spletni vmesnik,
- MySQL urejevalnik,
- uvoz CSV datotek,
- izvoz v različnih formatov,
- dodajanje kompleksnih poizvedb s pomočjo primerov (QBE - Querry-by-Example),
- podpora za vse operacijske sisteme.

2.2.6 Verzioniziranje programske kode

Verzioniranje programske kode je zelo pomembno pri projektu, ki se razvija v skupini. Pri diplomski nalogi nimamo skupine, ampak želimo si, da bi imeli zgodovino sprememb. Pri našem projektu se je to izkazalo za zelo uporabno. Verzioniranje je shranjevanje sprememb podatkov. Temelji na funkciji "revision level". Vsaka nova sprememba se shrani kot nova labela originalnega dokumenta. Najpomembnejša lastnost tega sistema je učinkovita obnovitev podatkov, če v kodi naredimo napako, jo lahko dokument povrnemo v stanje pred spremembo. Uporabili smo vgrajeni vtičnik v orodju Netbeans in spletni servis Bitbucket [28], ki smo jih med seboj povezali s povezavo do spletnega servisa.

Poglavje 3

Razvoj sistema

V poglavju bomo predstavili postopno izdelavo koncepta IS za rezervacije, ki bo obsegal vse zgoraj uporabljene tehnologije in njeno uporabo v diplomskem delu.

3.1 Arhitektura sistema

Sistem vsebuje tri večje module. Najpomembnejši del je aplikacijski strežnik, ki bo deloval kot spletni servis. Nato sledita mobilna in spletna aplikacija, ki sta povezani na servis. Delujeta tako, da pošiljata zahtevke in poizvedbe po določenih podatkih, ki jih strežnik sprejme. Odgovor strežnik posreduje kot podatke, ki so strukturirani v formatu JSON. Takšna arhitektura nam omogoča podporo za več različnih sistemov, ker je logika ločena od aplikacij. Zagotavlja nam večjo neodvisnost in možnosti za nadaljnji razvoj. Slika 2.2 prikazuje arhitekturno zasnovo IS.

3.1.1 Primeri uporabe

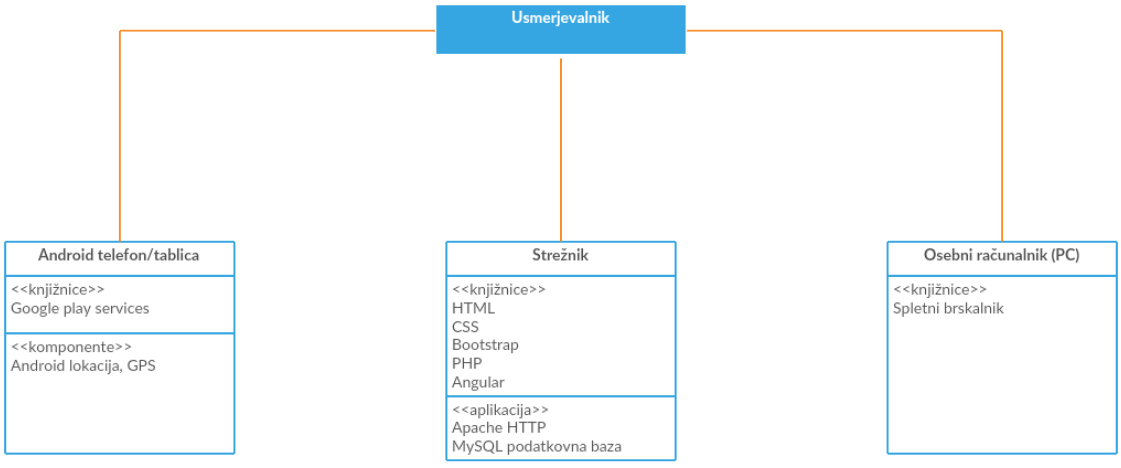
Glavni akterji sistema so uporabnik, upravljavec restavracije in administrator, ki ima vse pravice. Vsi akterji lahko dostopajo do obeh aplikacij, vse naprednejše funkcije sistema se izvajajo na spletni aplikaciji. Uporabnik lahko dodaja rezervacije, išče proste termine, ureja profil. Lastnik ima najpo-

membnejše delo, najprej mora ustvariti restavracijo, kateri doda mize, naslov in osnovne podatke o restavraciji. Zatem mora ustvariti še meni iz določenih jedi. Za registracijo jedi skrbi obrazec, ki ga lastnik izpolni, nato pa se odloči če bo le-ta objavljen v meniju ali ne. Administrator je upravljavec sistema, ki ima pravice za urejanje vseh podatkov, ki so v shranjeni v podatkovnem skladišču.

3.1.2 Implementacijski pogled

Na sliki (Slika 3.1) je diagram ključnih delov, ki jih sistem uporablja za svoje delovanje. Mobilna aplikacija za potrebe lokacije uporablja lokacijo, ki jo pridobi iz naprave (GPS ali lokacijske storitve). Za komunikacijo s spletnim servisom uporablja protokol HTTP, tako da strežnik s strani odjemalca prejme zahteve, te obdela ter jih nato pošlje nazaj napravi.

Za dostop do spletne aplikacije uporabljamo osebni računalnik ali pametni telefon, ki ima nameščen spletni brskalnik.



Slika 3.1: Implementacijski pogled

3.1.3 Podatkovni model

Za vsakim spletnim servisom stoji ustrezna shramba, ki je z njim povezana in je zelo pomembna za delovanje celotnega sistema. Zato je zahteva po učinkovitem in predvsem optimiziranem sistemu shranjevanja podatkov nujna. Pri tem projektu smo se odločili, da bomo uporabili relacijsko podatkovno bazo. Na začetku vsakega načrtovanja podatkovne baze moramo zgraditi podatkovni model. Z ustrezno načrtovanim podatkovnim modelom je mogoč tudi nadaljnji razvoj, kar pomeni, da je začetni korak zasnove zelo pomemben. Ker moramo zagotoviti transakcije, smo izbrali podatkovni motor InnoDB. V naslednjem primeru imamo dvanajst začetnih entitet, v katerih so shranjeni podatki o uporabnikih, lastnikih, rezervacijah, restavracijah in urniku (Slika 3.2). Vsi primarni ključi se z novimi vnosi samodejno povečujejo.

Entiteta "Oseba" hrani podatke o univerzalni osebi, ki je lahko lastnik ali uporabnik, kar se loči po drugih entitetah, ki so povezane. To so elektronski naslov, ime, priimek, datum rojstva, telefon, spol, sol in geslo za dostop do aplikacije, ki skrbita za varnost pred vdori v sistem. Entiteta je povezana še s tremi entitetami: "Uporabnik", "Lastnik" in "Naslov". Entiteta ima tuji ključ, ki ga pridobi iz entitete "Naslov".

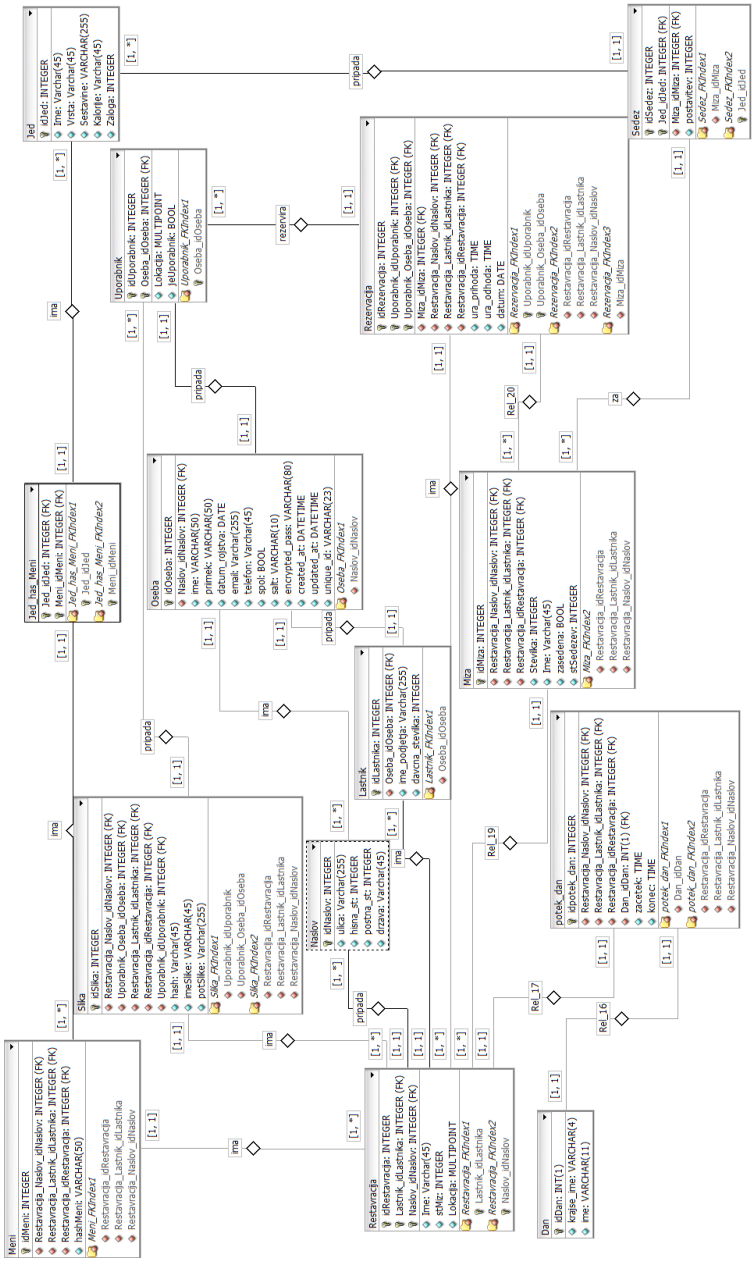
Entiteta "Uporabnik" hrani lokacijo uporabnika in vrednost ali je uporabnik ali ni. Entiteta je povezana z entitetama "Oseba" in "Rezervacija". Ima tuji ključ, ki ga pridobi iz entitete "Oseba".

Entiteta "Lastnik" hrani podatke o imenu podjetja in davčno številko. Entiteta je povezana z entitetama "Oseba" in "Restavracija". Ima tuji ključ, ki ga pridobi iz entitete "Oseba".

Entiteta "Naslov" hrani podatke o ulici, hišni številki, poštni številki, kraju in državi. Entiteta je povezana z entitetama "Oseba" in "Restavracija".

Entiteta "Dan" je statična podporna entiteta, ki vsebuje ime in krajše ime vsakega dneva v tednu. Entiteta je povezana z entiteto "potek_dan".

Entiteta "Restavracija" hrani podatke o imenu, številu miz in lokaciji restavracije. Entiteta je povezana z entitetami "Lastnik", "Naslov", "po-



Slika 3.2: Podatkovni model

tek_dan", "Meni", "Slika", "Rezervacija", "Miza". To je ena najpomembnejših entitet, ki jih bomo uporabljali v naši aplikaciji, saj povezuje vse po-

trebne podatke za registracijo rezervacije. Ima dva tuja ključa, ki jih pridobi iz entitet "Lastnik" in "Naslov".

Entiteta "Miza" hrani podatke o številki, imenu, številu sedežev in podatku ali je zasedena ali ne. Entiteta je povezana z entitetami "Restavracija", "Sedez", "Rezervacija". Ima tri tuje ključe, ki jih pridobi iz entitete "Restavracija".

Entiteta "potek_dan" hrani podatke o začetku in koncu dneva, uporabljamo jo za shranjevanje urnikov po dnevih. Entiteta je povezana z entitetama "Dan" in "Restavracija". Ima štiri tuje ključe, prve tri pridobi iz entitete "Restavracija" preostali ključ pa iz entitete "Dan".

Entiteta "Sedez", hrani podatke postavitvi sedeža za mizo. Entiteta je povezana z entitetama "Miza" in "Jed". Ima dva tuja ključa, ki jih pridobi iz entitet "Miza" in "Jed".

Entiteta "Meni"hrani hash menija. Entiteta je povezana z entitetama "Restavracija" in "Jed", ki je povezana s povezavo n,m, ta zahteva dodatno vmesno tabelo "jed_has_meni". Ima tri tuje ključe, ki jih pridobi iz entitete "Restavracija".

Entiteta "Jed" hrani podatke o jedi: ime, vrsta, sestavine, kalorije, zaloga. Entiteta je povezana z entitetama "Sedez" in "Meni", ki je povezana s povezavo n,m, spet se ustvari vmesna tabela "jed_has_meni".

Entiteta "Slika" hrani "hash", ki se ustvari ob dodajanju slike, ime slike in pot do slike. Posamezne slike so shranjene v svojih mapah, ki imajo v naslovu mape hash vrednost, ta pa enolično definira sliko. Entiteta je povezana z entitetami "Uporabnik" in "Restavracija'. Ima pet javnih ključev, ki jih pridobi iz entitet "Uporabnik" (dve) in "Restavracija" (tri).

Po izdelanem fizičnem modelu ER (Slika 3.2) lahko ustvarimo datoteko SQL, ki kreira tabele in vse relacije oziroma povezave med njimi, kot pomožna (statična) je dodana tabela Seznam poštnih številk z imeni. Podatkovno bazo tako sestavlja petnajst tabel (Slika 3.3). Spodnji primer kode prikazuje stavek MySQL CREATE.

	Tabela	Dejanje	Vrstic	Vrsta	Pravilo za razvrščanje znakov	Velikost	Presežek
☐	dan	Prebrskaj Struktura Iskanje Vstavi Izprazni Zavrzi	7	InnoDB	utf8_bin	16 KiB	-
☐	jed	Prebrskaj Struktura Iskanje Vstavi Izprazni Zavrzi	0	InnoDB	utf8_bin	16 KiB	-
☐	jed_has_meni	Prebrskaj Struktura Iskanje Vstavi Izprazni Zavrzi	0	InnoDB	utf8_bin	48 KiB	-
☐	lastnik	Prebrskaj Struktura Iskanje Vstavi Izprazni Zavrzi	1	InnoDB	utf8_bin	32 KiB	-
☐	meni	Prebrskaj Struktura Iskanje Vstavi Izprazni Zavrzi	0	InnoDB	utf8_bin	32 KiB	-
☐	miza	Prebrskaj Struktura Iskanje Vstavi Izprazni Zavrzi	0	InnoDB	utf8_bin	32 KiB	-
☐	naslov	Prebrskaj Struktura Iskanje Vstavi Izprazni Zavrzi	1	InnoDB	utf8_bin	16 KiB	-
☐	oseba	Prebrskaj Struktura Iskanje Vstavi Izprazni Zavrzi	7	InnoDB	utf8_bin	32 KiB	-
☐	potek_dan	Prebrskaj Struktura Iskanje Vstavi Izprazni Zavrzi	21	InnoDB	utf8_bin	64 KiB	-
☐	restavracija	Prebrskaj Struktura Iskanje Vstavi Izprazni Zavrzi	1	InnoDB	utf8_bin	48 KiB	-
☐	rezervacija	Prebrskaj Struktura Iskanje Vstavi Izprazni Zavrzi	0	InnoDB	utf8_bin	64 KiB	-
☐	sedez	Prebrskaj Struktura Iskanje Vstavi Izprazni Zavrzi	0	InnoDB	utf8_bin	48 KiB	-
☐	seznam_postni	Prebrskaj Struktura Iskanje Vstavi Izprazni Zavrzi	494	InnoDB	utf8_bin	48 KiB	-
☐	slika	Prebrskaj Struktura Iskanje Vstavi Izprazni Zavrzi	0	InnoDB	utf8_bin	48 KiB	-
☐	uporabnik	Prebrskaj Struktura Iskanje Vstavi Izprazni Zavrzi	5	InnoDB	utf8_bin	32 KiB	-
	15 tabel	Vsota	537	InnoDB	utf8_bin	576 KiB	0 B

Slika 3.3: Tabele podatkovnega modela

```
/*Primer MySQL CREATE stavka*/
CREATE TABLE Oseba (
  idOseba INTEGER NOT NULL AUTO_INCREMENT,
  Naslov_idNaslov INTEGER NULL,
  ime VARCHAR(50) NOT NULL,
  priimek VARCHAR(50) NOT NULL,
  datum_rojstva DATE NULL,
  email Varchar(255) NOT NULL,
  telefon Varchar(45) NULL,
  spol BOOLEAN NULL,
  salt VARCHAR(10) NOT NULL,
  encrypted_pass VARCHAR(80) NOT NULL,
  created_at DATETIME NOT NULL,
  updated_at DATETIME NULL,
  unique_id VARCHAR(23) NOT NULL,
  PRIMARY KEY(idOseba),
  INDEX Oseba_FKIndex1(Naslov_idNaslov)
)
```

```
TYPE=InnoDB;
```

3.2 Spletna aplikacija

Pred izdelavo mobilne aplikacije moramo narediti spletni del aplikacije. Spletni del obsega registracijo lastnika, uporabnika, restavracije, urnika, mize in jedi. Pomembno pri tem je, da so aplikacije na pogled podobne, kar bomo dosegli tako, da bo imela spletna aplikacija podobne poglede kot mobilna. To so iskanje primerne restavracije z mizo, ki je prosta za dobo rezervacije, ter rezervacijo le te.

3.2.1 Uporabniški vmesnik

Projekt je zgrajen na dveh zelo različnih osnovah, na eni strani je mobilna aplikacija, na drugi pa spletni portal. Vsaka aplikacija zase ima svoje oblikovne in funkcionalne zahteve. V tem poglavju se bomo srečali z oblikovnimi zahtevami. Mobilna aplikacija mora biti enostavna in pregledna, ker smo zelo omejeni s prostorom. Spletni brskalnik na drugi strani pa ni tako omejen s prostorom, saj je delovna površina neprimerno večja, zato lahko naredimo naprednejše in bolj podrobne aplikacije. Vključimo lahko vse funkcionalnosti aplikacije.

Grafični uporabniški vmesnik (angl. *GUI*) je orodje, ki prikazuje naslednje elemente: ikone, okna in animacije. Temelji zanj so bili postavljeni v raziskovalnem oddelku Xerox PARC leta 1973 [29].

Vmesnik je zgrajen na osnovi Bootstrapa. Kot smo že spoznali je ogrodje namenjeno dinamičnim spletnim stranem, ki s pomočjo AngularJS tvorijo celotno spletno aplikacijo - strokovno aplikacijski del "front end". Ta del aplikacije je namenjen izključno uporabniškemu vmesniku aplikacije in je tudi s kodo viden končnemu uporabniku. Prednost AngularJS je v tem, da lahko v realnem času poizvedujemo po podatkovni bazi in s tem osvežujemo samo tisti del, ki je potreben. To je najbolj pomembno pri načrtovanju

uporabniku prijaznega obrazca za vnos podatkov. Uporabnika lahko s to tehnologijo sproti obveščamo o napaki pri vnosu podatkov.

3.2.1.1 Razvoj uporabniškega vmesnika

Za razvoj potrebujemo več tehnologij, ki skupno povezujejo celoto strani. Za grafično podobo skrbi datoteka CSS, kjer sta opisana tako postavitev, kot podoba strani. HTML uporabljamo kot nosilec vseh elementov, ki so povezani v celoto. Tukaj dodajamo vse datoteke CSS in datoteke Javacript, na katerih temelji AngularJS, Bootstrap in nekatere knjižnice, ki jih bomo uporabili. Ko so vse tehnologije dodane na stran, lahko začnemo z razvojem rešitve. CSS in AngularJS imata klice funkcij v datoteki HTML. AngularJS ima lepo lastnost in menim, da je to največja prednost pred drugimi knjižnicami, da lahko tudi s klicem v datoteki HTML pišemo enostavnejšo kodo. S tem je pisanje kode olajšano in bolj čisto v primerjavi z jQuery in podobnimi knjižnicami.

```
// Primer HTML kode, ki vsebuje vse tri elemente, ki smo jih
    zgoraj opisali (HTML, CSS, AngularJS)
<div ng-class="{ 'has-error' : errorcasRez }">
   <select name="subjectList" ng-options="n for n in [] |
       range1:30:81" class="form-control"
       ng-model="formData.casRez" required>
      <option value="">Izberi cas rezervacije</option>
   </select>
   <span class="help-block" ng-show="errorcasRez">{{ errorcasRez
       }}</span>
</div>
```

Na zgornjem primeru je lepo razvidno, kaj točno kličemo s CSS na primer class='help-block', skrbi za pravilno strukturo in prikaz obveščanja pri napaki. DIV in SELECT so elementi HTML. Tehnologijo AngularJS vsebujejo vsi elementi v strukturi HTML, ki se začnejo z "ng", te kličejo AngularJS aplikacijo. Z "ng-options"pokličemo "range1"funkcijo v dodatni datoteki, ki

vsebuje funkcije in kontrolerje za popoln nadzor nad obrazcem. Funkcija je prikazana v spodnjem primeru. V našem primeru se sprehodimo po izbirah in jih zapišemo v tabelo, nato AngularJS poskrbi, da s pomočjo pretvorbe dobimo HTML izbire v SELECT elementu, kot jih potrebujemo v naši spletni formi. Zanimivo je, da v praksi to deluje zelo hitro.

Opisali smo postopek s strani HTMLja, zdaj moramo pogledati, kako se stvari streže v AngularJS svetu. Kot smo že prej opisali AngularJS, deluje na obeh straneh tako na HTML, kot na JS strani. Spodaj si bomo pogledali primer, kako delujejo funkcije v Angularju.

```
// Primer AngularJS funkcije
formApp.filter('range1', function() {
  return function(input, min, max) {
    min = parseInt(min);
    max = parseInt(max);
    for (var i=min; i<max; i=i+10)
      input.push(i+":"+0+0);
    return input;
  };
});
```

Primer preproste funkcije, ki smo jo uporabili pri prejšnjem primeru. Torej stvari delujejo po principu klicanja funkcij in podajanja parametrov. Podobno kot to delamo pri Javi. Čeprav ima le-ta veliko lastnosti Javascripta se po klicu in uporabi funkcij razlikujeta. Ena najpomembnejših lastnosti je, da ima AngularJS validacijo obrazcev, ki jo uporabimo. V prejšnjem primeru, smo spoznali, kako uporaba funkcij olajša klic zapletenih elementov strani, zdaj lahko podobno vpeljemo za uporabno validacijo obrazcev. Tako pokličemo določen model *ng* in dobimo podatke v realnem času za vnos, ki ga lahko s pomočjo Javascripta preverimo, kakor želimo. Če želimo vnos preveriti na strežniku ali želimo vnesti podatke v podatkovno bazo, moramo preko POST protokola in s pomočjo JSON-a komunicirati s PHP stranjo. Od tam naprej pa s pomočjo PHP funkcij, ki so zapisane v razredu *DB_Functions*

komuniciramo s podatkovno bazo. Podatke smo tako lahko preverili ali vpisali v podatkovno bazo. Ta postopek bomo imenovali "Spletna povezava podatkovne baze". Funkcije, ki jih bomo uporabljali, so iste tudi za mobilni del aplikacije, ki ga bomo predstavili v naslednjem podpoglavju "Android aplikacija".

3.2.2 Implementacija funkcij (back-end)

Pretežni del implementacije opravimo z razvojem funkcij, ki nam pomagajo ustvariti povezavo med bazo podatkov in PHP stranjo. Delujejo po principu pripravljenih funkcij, ki jih potem v sami aplikaciji kličemo s podatki. Glede na to, da je naša aplikacija tako obsežna smo se odločili na primeru predstaviti, kako ustvarimo funkcijo na določeno tabelo v bazi. Pri prijavi moramo vsakega uporabnika preveriti. V spodnjem primeru je pokazano, kako to storimo v razredu *DB_Functions*.

```
//Primer uporabe DB funkcjie
public function getUserByEmailAndPassword($email, $password) {
        $stmt = $this->conn->prepare("SELECT * FROM oseba WHERE
            email = ?");
        $stmt->bind_param("s", $email);
        if ($stmt->execute()) {
            $user = $stmt->get_result()->fetch_assoc();
            $stmt->close();
            $salt=$user['salt'];
            $pass=$user['encrypted_pass'];
            if($this->checkhashSSHA($salt, $password)== $pass){
                return $user;
            }else{
                return null;
            }
        } else {
            return NULL;
```

```
        }
}
```

Uporaba mysqli stavkov nam omogoča, da enostavno in pregledno pridobimo in preverimo podatke.

Kako poteka klic funkcij v razredu *DB_Functions*? V stran PHP dodamo razred *DB_Functions* in deklariramo nov objekt "db", ki ga bomo uporabljali pri preverjanju uporabnika.

```
//Primer klica funkcije
require_once 'include/DB_Functions.php';
$db = new DB_Functions();
//Parametri za preverjanje preko POST metode
$email = $_POST['email'];
$password = $_POST['password'];
//uporabimo funkcijo in pridobimo uporabniske podatke
$user = $db->getUserByEmailAndPassword($email, $password);
```

S funkcijami in njihovo uporabo nastane lep model za ustrezno in učinkovito razvijanje aplikacije v prihodnosti. Za učinkovit razvoj aplikacije potrebujemo vsaj še enega razvijalca, ki bi se s pomočjo modela hitro vključil v razvoj sistema.

3.2.3 Rezervacija

Zaradi obsežnosti aplikacije ne bomo razlagali registracij restavracije in vse, kar spada pod domeno lastnika. Vseeno moramo nekaj omeniti o samem postopku registracije rezervacije. Najprej moramo poiskati rezervacijo, ki je prosta v želenem terminu. Vstopna stran nas pozdravi z našo približno lokacijo, če ne želimo dovoliti brskalniku, da za nas sam pridobi lokacijo, pa jo moramo vpisati sami. Iskanje je preprosto saj v obrazec, ki leži nad zemljevidom, vpišemo število oseb, datum in uro rezervacije ter čas rezervacije. Izpiše se nam seznam restavracij z možnimi termini. Ob kliku na

termin sistem samodejno izbere prosto mizo za izbor jedi. Izbor jedi poteka tako, da se nam izpišejo stoli za vsako osebo posebej, nato izberemo iz menija, ki se nam odpre nad stranjo želeno jed. Postopek ponovimo za vse osebe. Klasična rezervacija je videti tako, da izberemo želeni termin po zgornjem postopku, v aplikaciji pa odkljukamo "brez rezervacije jedi", stoli se nam ne prikažejo. Vse rezervacije so vidne tako uporabnikom, kot lastnikom zato so v trenutku potrditve rezervacije potrjene s strani uporabnika in lastnika. Rezervacija postane v istem trenutku zavezujoča za oba akterja. V konceptu ni predstavljen sistem, ki bi lahko kaznoval uporabnika, ki ne pride na rezervacijo.

Razmišljali smo, da bi se uvedel sistem potrjevanja rezervacij. Če uporabnik svoje rezervacije ne unovči, dobi po poteku rezervacije avtomatsko opozorilo. S ponavljajočimi se opozorili lahko uporabnik izgubi dostop do strani. Preverjanje bi delovalo po posebni QR kodi, ki bi bila unikatna na vsak izdani račun in bi jo uporabnik s pomočjo mobilne aplikacije vpisal na svoj račun. Uporabniki, ki ne morejo kode potrditi z mobilno aplikacijo, bi bila koda tudi v obliki črk in številk, za kasnejši vnos v spletno aplikacijo. Nekatere restavracije nimajo računalniškega sistema in pišejo račune na roke. V takem primeru bi imeli posebno enkratno kodo, ki bi jo uporabnik dobil skupaj z na roke napisanim računom. Podoben sistem bi uvedli tudi na strani lastnikov, prav tako bi imeli opozorila za počasno postrežbo ali nekakovostno prehrano, ki bi bila regulirana s strani moderatorja. Restavracijo, ki ima več opozoril, bi blokirali, da ne bi mogla več sprejemati rezervacij. Za ponovno registracijo restavracije bi morali lastniki predložiti ustrezne ukrepe, s katerimi bi izboljšali izdajo in kakovost jedi. Tako bi se izognili neprihodom izbrane rezervacije, slabe kakovosti prehrane in predvsem, kar je najpomembneje počasne izdaje prehrane. Glede na opozorila bi lahko ustvarili rang tako restavracij kot uporabnikov. Restavracije bi se lahko odločile, da tega sistema ne bodo uporabljale, v tem primeru, bi jim bila onemogočena vnaprejšnja rezervacija hrane.

3.3 Android aplikacija

V tem poglavju bomo predstavili potek izdelave mobilne aplikacije. Aplikacija bo bazirala na sistemu Android. Predstavljena bo tudi povezava s spletnim servisom.

3.3.1 Nastavitve AndroidManifest in Gradle datotek

Android Manifest (AndroidManifest.xml) je datoteka, ki skrbi, da v Androidu nastavimo vse osnovne nastavitve aplikacije. V njej določimo minimalno in ciljno razvijalno različico, ime aplikacije, podobo ikone, ki se izriše na začetnem zaslonu sistema ter dovoljenja aplikacije. V datoteki navedemo vse aktivnosti, ki jih bomo v aplikaciji izvajali. Za komunikacijo s spletnim servisom, lokacijske storitve aplikacije in dostop do medomrežja potrebuje dovoljenja, ki jih omogočimo s spodnjo kodo:

```
<uses-permission
   android:name="android.permission.INTERNET" />
<uses-permission
    android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission
    android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission
    android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

Gradle (build.gradle) je datoteka, ki skrbi za uvoz različnih knjižnic. Za aplikacijo smo uporabili naslednje:

```
    compile 'com.android.support:appcompat-v7:23.1.1'
    compile 'com.android.support:support-v4:23.1.1'
    compile 'com.mcxiaoke.volley:library-aar:1.0.0'
    compile 'com.google.android.gms:play-services:8.3.0'
```

3.3.2 Mobilni uporabniški vmesnik

Za razvoj uporabniških vmesnikov Android uporabljamo označevalni jezik XML, s katerim smo vsaki aktivnosti ali fragmentu namenili XML datoteko, ki definira podobo zaslona. Elemente lahko dodajamo tudi programsko preko kode, če potrebujemo dinamično se spreminjajoče elemente, moramo uporabiti tako metodo. Neposredno prek ukazov v programskem jeziku Java, dodajamo določene elemente na zaslon.

```
//Primer dodajanja elementa na platno
Button element = new Button(this);
element.setText(getResources.getText(R.string.element));
element.setLayoutParams(new
    LayoutParams(AbsListView.LayoutParams.FILL_PARENT,
AbsListView.LayoutParams.WRAP_CONTENT));
linerLayout.addView(element);
```

3.3.3 Aktivnosti in fragmenti

Aktivnost si lahko predstavljamo kot razred pri Javi za delovanje in funkcionalnost aktivnosti, ki ima tudi XML datoteko za podobo. Da jo lahko uporabljamo, jo moramo deklarirati v datoteki AndroidManifest, kot je prikazano v spodnjem primeru.

```
// Primer deklaracije v datoteki AndroidManifest
<activity
            android:name=".LoginActivity"
            android:label="@string/app_name"
            android:launchMode="singleTop"
            android:windowSoftInputMode="adjustPan">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
                    android:name="android.intent.category.LAUNCHER"
```

```
                />
            </intent-filter>
 </activity>
```

Aktivnost ima tudi svoje fragmente, ki jih uporabljamo med izvajanjem. Prav tako jih sestavljata XML in javanska datoteka. Znotraj aktivnosti lahko gostuje več fragmentov, ki med seboj komunicirajo.

Slika 3.4: Predstavitev fragmenta

Mobilna aplikacija vsebuje več aktivnosti, zato moramo v datoteki AndroidManifest določiti aktivnost, ki se ob zagonu izvede prva, to je tudi opisano v zgornjem primeru. "LoginActivity" se izvede prva in takoj ob zagonu preveri, ali ima veljavno sejo. V primeru pozitivnega odgovora aktivnost preusmeri na "MainActivity", kjer lahko preko menija izbiramo vse aktivnosti, ki smo jih povezali nato nas aplikacija, preusmeri na izbrano aktivnost. V nadaljevanju bomo te aktivnosti opisali in predstavili.

Ena od lastnosti fragmentov je tudi ta, da lahko na velikem zaslonu predstavimo oba fragmenta skupaj, lahko pa ju razdelimo na dva zaslona, če imamo manjši zaslon, kar lepo vidimo na zgornji sliki 3.4.

3.3.3.1 Aktivnost LoginActivity

Ta aktivnost se uporablja s prvim zagonom aplikacije. Tu preverimo, ali je uporabniška seja veljavna, ali ne. Če je seja veljavna se aktivnost preskoči

in nadaljuje na "MainActivity", ki skrbi za drsni meni in meni aktivnosti, preko katerih preko servisa interaktivno obdelujemo in shranjujemo podatke.

```
// Primer deklaracije v AndroidManifest datoteki
if (session.isLoggedIn()) {
            // User is already logged in. Take him to main activity
            Intent intent = new Intent(LoginActivity.this,
                MainActivity.class);
            startActivity(intent);
            finish();
        }
```

V primeru neveljavne seje nas aplikacija preusmeri na mehanizem za preverjanje prijave uporabnika, ki deluje tako, da v vnosne dialoge vpišemo uporabniško ime in geslo. S POST metodo na servisu preverimo podatke, servis nam vrne podatke v JSON formatu (spodnji primer), ki jih potem z GSON knjižnico pretvorimo objekt.

```
//Primer JSON odgovora
{"error":false,"uid":"56616c4a47c706.91670851",
"user":{"ime":"Milce","email":"milce.hribov@gmail.com",
"created_at":"2015-12-04 11:34:50","updated_at":null},
"lastnik":{"davcna_stevilka":298491}}
```

Ko dobimo odgovor strežnika, je zelo pomembno, da poznamo strukturo podatkov, ker na podlagi tega zgradimo celotno logiko aplikacije. V primeru odgovora je na začetku atribut "error" postavljen na "false", to pomeni da smo dobili pozitiven odgovor iz strežnika, moramo zdaj pretvoriti vsebino, ki se nahaja v atributu "user" v SQLite podatkovno bazo, ki nam kasneje služi za shranjevanje seje uporabnika, da se mu ni treba vedno znova prijavljati ob zagonu aplikacije. Na spodnjem primeru si bomo pogledali, kako to deluje v praksi.

```
//Primer dodajanja uporabnika
    public void addUser(String name, String email, String uid,
```

```
        String created_at) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(KEY_NAME, name); // Name
        long id = db.insert(TABLE_USER, null, values);
        db.close();
    }
```

Zgornji primer lahko razširimo na večjo podatkovno zbirko in lahko prav vse iz servisa shranimo po istem postopku. Dobili smo objekte s podatki. Za urejanje dodamo novo datoteko na spletni servis z UPDATE sql stavkom in jo kličemo znotraj aktivnosti s POST zahtevkom. Če je, zahtevek uspešen dobimo potrditveni odgovor, tako dobimo celotni paket za povezavo z bazo. Za brisanje bi naredili na primeru zgornjega postopka, samo na spletnem servisu dodamo DELETE ali INSERT namesto UPDATE stavek. Za lažjo referenco bomo ta postopek poimenovali "Android povezava z bazo".

3.3.3.2 Aktivnost MainActivity

Aktivnost vsebuje povezave do vseh funkcij programa. V tej aktivnosti nas pozdravi naša lokacija z bližnjimi restavracijami, ki jih dobimo iz servisa in vnosno polje, ki skrbi za vnos časa. V primeru, da ne odkljukamo, da dovolimo lokacijo, moramo preusmeriti aktivnost na drugi fragment, ki nas najprej vpraša za naslov, šele nato se nam prikaže lokacija z vpisanim naslovom in bližnjimi restavracijami. Zgodba mobilne aplikacije nas pelje v dodatno možnost, ki zahteva prisotnost našega sistema in lokacijo, da nam na zemljevidu izriše samo tiste restavracije, ki so za nas zanimive tisti trenutek, ko smo na poti.

MainActivity potrebuje Google maps knjižnico, s pomočjo katere dobimo lokacijo. Z Androidom 6.0 smo dobili novo možnost uporabniškega dodeljevanja pravic, zato moramo tudi programsko poskrbeti, da preverimo, ali je pravica dovoljena ali ni. Če ni, moramo vprašati uporabnika, ali jo dodeli ali ne. V naslednjem primeru je prikazana logika preverjanja dovoljenja.

```
//Primer preverjanja ali je Android 6.0
if(android.os.Build.VERSION.SDK_INT <
    android.os.Build.VERSION_CODES.M){
            imaDovoljenje=true;
            Intent intent = new Intent(MainActivity.this,
                MapsActivity.class);
            startActivity(intent);
            finish();
        }else{
            handlePermissionsAndGetLocation();
        }
```

Preverili smo različico programa, če imamo Android, ki ustreza kriterijem preverimo dovoljenje za lokacijo, če je le-ta potrjena pridobimo lokacijo.

```
//Primer preverjanja dovoljenja lokacije
if (ActivityCompat.checkSelfPermission(this,
    Manifest.permission.ACCESS_FINE_LOCATION)
    == PackageManager.PERMISSION_GRANTED){
      myLocation = locationManager.getLastKnownLocation(provider);
}
```

Dovoljenja so preverjena, nato moramo dobiti podatke o restavracijah in jih prikazati glede na njihovo lego. V podatkovni bazi ima restavracija namreč zapisan podatek o lokaciji, ki vsebuje koordinate objekta. S stavkom SELECT izberemo vse restavracije znotraj območja petih kilometrov:

```
#(statX, statY) - Primarni koordinati, ki jih pridobimo iz
    aktivnosti
SELECT *
FROM Restavracija
WHERE acos(sin(statX) * sin(X(Restavracija.Lokacija)) + cos(statX)
    * cos(X(Restavracija.Lokacija)) * cos(Y(Restavracija.Lokacija)
    - (statY))) * 6371 <= 5;
```

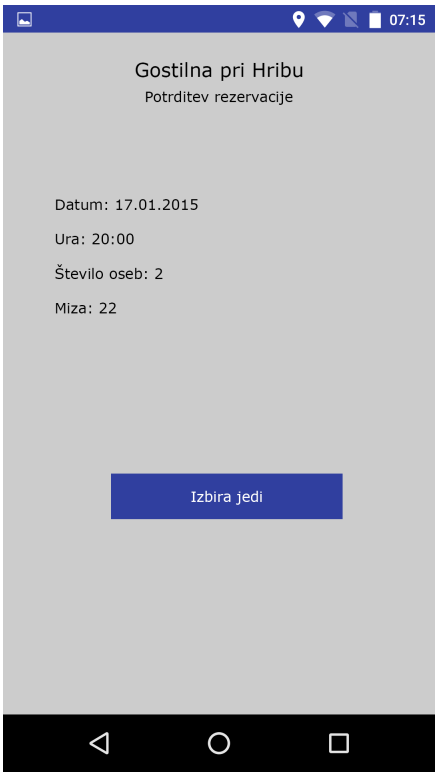
Dobili smo točke za prikaz restavracij na zemljevidu, ki smo jih okoli primarnih koordinat pridobili iz aplikacije. Ker izberemo najbližje restavracije, s tem uporabniku zagotovimo hitrejši dostop do želene restavracije. Zato je iskanje uporabniku preprostejše in uporabnejše od navadnega iskanja. Hkrati moramo preveriti še, ali imamo prost termin za vpisan čas. Vse podatke si izmenjujemo s pomočjo POST zahtevka, ki je opisan v postopku "Android povezava z bazo".

V zgornjem kotu na levi strani lahko kliknemo za dostop do drsnega menija, ki vsebuje vse uporabniške aktivnosti za lažjo navigacijo. Drsni meni je zelo praktičen, saj nam ni potrebno spreminjati zaslona medtem ko hočemo na drugi fragment ali aktivnost. Tako je naša aktivnost zaključena (Slika 3.5).

Slika 3.5: Prikaz izgleda MainActivity.

3.3.3.3 Aktivnost ReservationActivity

Enostavno in hitro delo z aplikacijo nam omogoča "MainActivity", ki nam izbere restavracijo iz zemljevida. Tehnologijo in razvoj smo opisali v prejšnjih aktivnostih, zato temu na tem mestu ne bomo posvečali pozornosti. Na to aktivnost smo preusmerjeni, ko na zemljevidu kliknemo na izbrano restavracijo. Odpre se nam fragment, ki tako kot pri spletni aplikaciji prikazuje nekaj dodatnih terminov. Ob kliku na termin se nam odpre obrazec z možnostjo izbire menija po istem postopku, kot je opisan v spletni aplikaciji. Potrdimo rezervacijo (Slika 3.6) in dodamo jedi v sistem. Ko smo rezervacijo zaključili nas sistem preusmeri na seznam aktivnih rezervacij.



Slika 3.6: Prikaz izgleda ReservationActivity.

Poglavje 4

Sklepne ugotovitve

Načrtovanje in izdelava diplomske naloge se mi je zdela zabavna in zanimiva. Naučil sem se veliko novih stvari. Od ideje do koncepta je vloženega veliko dela in znanja. Če začnemo pri podatkovnem modelu, odločitev za neko tehnologijo ni preprosta, potrebno je namreč raziskati vse razpoložljive rešitve in glede na zahteve in zmožnosti strojne opreme izbrati tisto, ki nam najbolj odgovarja. Pri izbiri tehnologij mi je bil v pomoč sistem za evidentiranje živine [8]. Za naš projekt obstajata dve zelo dobri rešitvi. MongoDB je podatkovna baza, ki ne temelji na konceptih relacijskega podatkovnega modela in je zato bolj primerna za veliko število podatkov. Na drugi strani pa je MySQL še vedno ena najpogostejših relacijskih podatkovnih baz in glede na moje pretekle izkušnje najbolj primerna za razvoj našega koncepta. Na začetku sem želel biti kar se da kompatibilen z današnjimi spletnimi gostovanji, ki ne podpirajo naprednih potreb stranke. Če se nam v prihodnosti pokaže potreba po večji in naprednejši podatkovni bazi, obstaja možnost migracije, prav tako bi bilo potrebno razmišljati o ustrezni rešitvi gostovanja. Seveda nam ob migraciji ne preostane drugega, kot da popravimo vse klice v vseh aplikacijah, vendar je to tveganje, ki sem ga pripravljen sprejeti. Pri spletni aplikaciji te dileme ni, saj je trenutno tehnologija, ki sem jo uporabil za naše zahteve najboljša možna. Tehnologija spletnih aplikacij se razvija s svetlobno hitrostjo, to narekuje konstanten razvoj sistema. Mobilna aplika-

cija ima veliko možnosti za razvoj. Izbral sem tehnologije, ki so preproste za razvoj in razumevanje, obenem pa nudijo veliko hitrost in prilagodljivost. Vesel sem, da lahko vse uporabljene tehnologije in pridobljeno znanje uporabim pri nadaljnjih projektih. Največji problem, ki sem ga zaznal pri izdelavi diplomskega dela, je nedvomno obsežnost aplikacije. Takoj za tem pa sistem za preverjanje rezervacije. Moja ideja je na tem področju edinstvena, saj take implementacije v moji predhodni raziskavi nisem zasledil. Uporabnike je potrebno privabiti na stran z unikatno ponudbo. Veliko težavo vidim v tem, da bi se uporabniki z dodatnim delom, ki ga veleva sistem za preverjanje rezervacije, zaradi tega odločali za klasične rezervacije. V končni verziji aplikacije, bi morali sistem dodobra testirati in če bi bilo potrebno prilagoditi uporabnikovim in lastnikovim željam.

4.1 Možnosti za nadaljnji razvoj

Koncept je pri koncu, pravo delo pa se s tem šele začne. Aplikacija ima pred seboj verjetno še eno leto razvoja in iskanja najboljše možne rešitve za potrditev rezervacije. Možnosti za izboljšave so na vseh področjih. Do končne izdaje sistema je potrebno govoriti z vsemi akterji aplikacije in tudi po njihovih željah dopolnjevati aplikacijo. Resno delo zahteva tudi pravo ekipo, ki zaenkrat še ne obstaja. Možno je povezovanje s podjetji, ki imajo podobne rešitve že razvite, v tem primeru si je potrebno pogledati tudi, kako pravno zaščiti idejo, saj menim da je komercialno zelo zanimiva.

Sami aplikaciji verjetno manjka še veliko do končnega produkta, a treba je imeti v mislih tudi razvoj različnih APIjev za integracijo s spletno stranjo, ki jo ima restavracija. Še ena velika prednost bi bila ponuditi API za povezave z obstoječimi informacijskimi sistemi. Za restavracije, ki informacijskega sistema še nimajo, bi bilo mogoče v sodelovanju z različnimi podjetji ponuditi produkt, ki bi v največji meri izkoristil prednosti našega rezervacijskega sistema. To je tudi končni cilj in gonilo za prihodnost. Končni sistem z vsemi funkcionalnostmi bi lahko imenovali "Pametne restavracije".

Literatura

[1] OpenTable, [Online]. Dosegljivo:
http://www.opentable.com/. [Dostopano 04.01.2016].

[2] dimmi, [Online]. Dosegljivo:
https://www.dimmi.com.au/. [Dostopano 09.02.2016].

[3] MyTable, [Online]. Dosegljivo:
https://www.mytable.org/. [Dostopano 09.02.2016].

[4] ZAGAT, [Online]. Dosegljivo:
https://www.zagat.com/. [Dostopano 09.02.2016].

[5] Rezervacije iz restavracije Manna, [Online]. Dosegljivo:
https://manna.youcanbook.me/. [Dostopano 17.01.2016].

[6] YOU CAN BOOK ME, [Online]. Dosegljivo:
https://youcanbook.me. [Dostopano 16.01.2016].

[7] FOOD WASTE FACTS, [Online]. Dosegljivo:
http://www.unep.org/wed/2013/quickfacts/.
[Dostopano 8.12.2015].

[8] A. Hafner (2015) Prototip sistema za evidentiranje živine, [Online]. Dosegljivo:
http://eprints.fri.uni-lj.si/3065/1/63100034-ANDREJ_HAFNER-Prototip_sistema_za_evidentiranje_%C5%BEivine.pdf.
[Dostopano 15.02.2016].

[9] JSON, [Online]. Dosegljivo:
https://en.wikipedia.org/wiki/JSON. [Dostopano 8.12.2015].

[10] JSON Array, [Online]. Dosegljivo:
http://www.w3schools.com/json/json_syntax.asp.
[Dostopano 8.12.2015].

[11] PHP, [Online]. Dosegljivo:
https://sl.wikipedia.org/wiki/PHP. [Dostopano 10.12.2015].

[12] Javascript, [Online]. Dosegljivo:
https://sl.wikipedia.org/wiki/JavaScript.
[Dostopano 10.12.2015].

[13] Java, [Online]. Dosegljivo:
https://en.wikipedia.org/wiki/Java_%28programming_language%
29. [Dostopano 10.12.2015].

[14] CSS, [Online]. Dosegljivo:
https://sl.wikipedia.org/wiki/CSS. [Dostopano 10.12.2015].

[15] Bootstrap (front-end framework), [Online]. Dosegljivo:
https://en.wikipedia.org/wiki/Bootstrap_%28front-end_
framework%29. [Dostopano 11.12.2015].

[16] Multi-screen guidelines, [Online]. Dosegljivo:
https://support.google.com/adsense/answer/6196932?hl=en.
[Dostopano 11.12.2015].

[17] MySQL, [Online]. Dosegljivo:
https://sl.wikipedia.org/wiki/MySQL. [Dostopano 11.12.2015].

[18] SQLite, [Online]. Dosegljivo:
https://en.wikipedia.org/wiki/SQLite. [Dostopano 11.12.2015].

[19] AngularJS, [Online]. Dosegljivo:
`https://en.wikipedia.org/wiki/AngularJS`.
[Dostopano 11.12.2015].

[20] Android Studio, [Online]. Dosegljivo:
`https://en.wikipedia.org/wiki/Android_Studio`.
[Dostopano 14.12.2015].

[21] Smartphone OS Market Share, 2015 Q2, [Online]. Dosegljivo:
`http://www.idc.com/prodserv/smartphone-os-market-share.jsp`.
[Dostopano 14.12.2015].

[22] Android (operating system), [Online]. Dosegljivo:
`https://en.wikipedia.org/wiki/Android_%28operating_system%29`. [Dostopano 14.12.2015].

[23] Android v prihodnosti hitrejši od iOS? ART - Andorid Runtime!, [Online]. Dosegljivo:
`https://gizzmo.si/blog/android-v-prihodnosti-hitreji-od-ios-art-andorid-runtime/`. [Dostopano 14.12.2015].

[24] Sam svoj programer, [Online]. Dosegljivo:
`http://www.monitor.si/clanek/sam-svoj-programer3/125026/`.
[Dostopano 14.12.2015].

[25] Dashboards, [Online]. Dosegljivo:
`http://developer.android.com/about/dashboards/index.html`.
[Dostopano 15.12.2015].

[26] phpMyAdmin, [Online]. Dosegljivo:
`https://en.wikipedia.org/wiki/PhpMyAdmin`.
[Dostopano 15.12.2015].

[27] Netbeans, [Online]. Dosegljivo:
`https://en.wikipedia.org/wiki/NetBeans`. [Dostopano 15.12.2015].

[28] Bitbucket, [Online]. Dosegljivo:
http://bitbucket.org/. [Dostopano 15.12.2015].

[29] GUI, [Online]. Dosegljivo:
https://sl.wikipedia.org/wiki/Grafi%C4%8Dni_uporabni%C5%
A1ki_vmesnik. [Dostopano 04.01.2016].